

Institut Préparatoire aux Études d'Ingénieurs de Tunis



Leila Ben Othman

Première année de classes préparatoires
MP-PC

Année universitaire: 2016 - 2017

Fonctions et procédures en algorithmique

Plan

① Programmation modulaire: Motivation et Intérêts

② Fonctions et procédures

- Fonctions

- Paramètres formels et effectif

- Structure d'un algorithme principal

- Procédures

- Mode de passage des paramètres

- Variables locales/globales

Motivation (1)

- ❶ Pour résoudre un problème, un algorithme effectue plusieurs tâches:
 - Saisie et contrôle de données
 - Recherche
 - Calcul
 - Affichage
 - ...
- ❷ De plus, le plus souvent, un algorithme procède à un même traitement (ou presque) à plusieurs endroits différents.

⇒ **Inconvénients:**

Algorithme lourd, illisible et qui contient des répétitions.

Motivation (2)

Solution: Décomposer le problème en sous-problèmes. On parle de **programmation modulaire**.

Un **module (sous-programme)** réalise une seule et unique tâche bien définie et bien délimitée.

Intérêts

- Répartir les difficultés: faciliter la résolution d'un problème complexe;
- Éviter les répétitions;
- Réutiliser le code;
- Faciliter la localisation des erreurs;

⇒ **Conséquences:**

Algorithme lisible, modulaire et facile à maintenir

Algorithme principal - Sous-programmes

- Le corps de l'algorithme est l'**algorithme principal**.
- Les sous-programmes auxquels on a recours s'appellent des **fonctions** et des **procédures**.
- L'algorithme principal sera constitué essentiellement d'**appels** aux sous-programmes.
- La communication est assurée à travers le passage de **paramètres** (**arguments**) et éventuellement un **retour** de résultat.

Les fonctions

Une fonction est un sous-programme contenant un certain nombre d'instructions et qui retourne un résultat.

Déclaration d'une fonction:

```
Fonction nom_fonction(paramètres formels: types) : type_résultat  
Var <Déclaration des variables locales>  
  Début  
    <Instructions>  
    retourner (résultat) ou bien nom_fonction ← résultat  
  Fin
```

Le type de la valeur retournée (résultat) doit correspondre à celui qui a été précisé dans l'entête de la définition de la fonction.

Exemple de fonction

```
fonction Somme(x,y : entier) : entier  
Var S: entier  
Début  
  S  $\leftarrow$  x+y  
  retourner(S)  
Fin
```

```
fonction Somme(x,y : entier) : entier  
Début  
  retourner(x+y)  
Fin
```

Appel de fonction

- ➊ Récupération du résultat de la fonction dans une variable

Variable $\leftarrow$ nom_fonction (paramètres effectifs)

- ➋ Affichage du résultat de la fonction

Écrire ("Le résultat est", nom_fonction(paramètres effectifs))

Paramètres formels et effectifs

On distingue deux types de paramètres :

- **Les paramètres formels:** sont les paramètres qui figurent dans l'entête de la définition de la fonction ou de la procédure.
- **Les paramètres effectifs:** sont les paramètres qui figurent dans l'instruction d'appel de la fonction ou de la procédure et sont substitués aux paramètres formels au moment de l'appel.

Remarque: Les paramètres effectifs et les paramètres formels doivent s'accorder du point de vue nombre et ordre ainsi que type.

⇒ On va voir que PYTHON est souple et flexible sur le typage des paramètres (typage dynamique), l'ordre des paramètres ainsi que leur nombre..

Structure d'un algorithme principal

Algorithme principal

Var

x,y,res: entier /* variables globales */

Fonction somme(a,b : entier) : entier /* Déclaration de la fonction */

Var

S: entier /* variable locale */

Début

S $\leftarrow$ a+b

retourner(S)

Fin

Début

Écrire(" Donner deux entiers")

Lire(x,y)

res $\leftarrow$ somme(x,y); /* Appel de la fonction */

Écrire("la somme est ", res)

Fin

Les procédures

Une procédure est un sous-programme contenant un certain nombre d'instructions mais qui n'admet pas de valeur de retour.

Déclaration d'une procédure:

```
Procédure nom_procédure(paramètres formels: types)
  Var <Déclaration des variables locales>
  Début
    <Instructions>
  Fin
```

L'appel d'une procédure se fait de la manière suivante:

```
nom_procédure(paramètres effectifs)
```

Exemple de procédure

Procédure somme_produit(x,y: entier, **var** s,p : entier)

Début

$$s \leftarrow x + y$$
$$p \leftarrow x^*y$$

Fin

- Le mot clef **var** désigne un mode de passage de paramètre par variable.

Structure d'un algorithme principal

Algorithme principal

Var

x,y,z,t: entier /* variables globales */

Procédure somme_produit(x,y, **var** s,p : entier) /* Déclaration de la Procédure */

Début

s ← a+b

p ← a*b

Fin

Début

Écrire(" Donner deux entiers")

Lire(x,y)

somme(x,y,z,t); /* Appel de la procédure */

Écrire("la somme est ", z)

Écrire("le produit est ", t)

Fin

Remarques

- Toute fonction doit comporter l'instruction retourner. Ce qui n'est pas le cas pour une procédure.
- La fonction calcule une valeur et la renvoie à l'algorithme principal.
- La procédure effectue un traitement, *i.e.*, elle ne renvoie rien de façon explicite comme le cas d'une fonction.
- Lorsque une procédure doit renvoyer un résultat vers l'algorithme principal. Ce résultat doit être transmis en tant que paramètre que la procédure peut modifier. On parle d'un paramètre avec un mode de passage par référence.
- Les paramètres que la procédure n'est pas censée modifier, doivent être passés par valeur.

Mode de passage des paramètres

- Ils existent deux modes de passage des paramètres : passage par valeur et passage par variable (appelé aussi par référence).
 - Le mode de passage par valeur: aucune modification n'est applicable sur ce paramètre. C'est donc le mode de passage des paramètres en entrée.
 - Le mode de passage par variable: modification possible sur ce paramètre. C'est donc le mode de passage des paramètres en sortie ou en entrée/sortie.
- Un paramètre transmis par variable est toujours précédé par le mot clé **var** dans l'entête de la définition de la procédure.
- Les paramètres d'une fonction sont toujours des paramètres en entrée et ils sont donc toujours passés par valeur.

Variables locales/globales

- Une variable locale est une variable définie au niveau d'une fonction/procédure. Elle n'est visible que dans cette fonction/procédure.
- Une variable globale est une variable définie au niveau de l'algorithme principal. Elle est donc visible par n'importe quelle fonction/procédure définie au niveau de cet algorithme.

⇒ On va revoir ces notions avec les fonctions PYTHON

De façon générale, il est déconseillé de faire usage de variables globales!!!